Modbus User Manual

Covers the following Modbus® RTU enabled drives:

ST5-Q-RN
ST5-Q-RE
ST5-Q-NN
ST5-Q-NE
ST10-Q-RN
ST10-Q-RE
ST10-Q-NN
ST10-Q-NE

STM17Q-3AN
STM17Q-3AE
STM17Q-3RN
STM17Q-3RE

STM23Q-2AN
STM23Q-2AE
STM23Q-2RN
STM23Q-2RE
STM23Q-3AN
STM23Q-3AE
STM23Q-3RN
STM23Q-3RE

STM24QF-3AN
STM24QF-3AE
STM24QF-3RN
STM24QF-3RE
SWM24QF-3AN
SWM24QF-3AE

APPLIED MOTION PRODUCTS, INC.

920-0072 Rev D
8/1/2014

What is Modbus RTU?

Modbus, originally created by Modicon, is a fieldbus that allows a master and one or more slave devices to share data. These data are organized into 16-bit registers, which can also be used to share information single-bit I/O points.

It is a popular protocol with PLC vendors due to its simplicity and the inherent ease of sending PLC register data (often 16-bits in width) over a fieldbus protocol optimized for 16-bit data.

The master may initiate read and write operations on single registers or blocks of registers. While there is no rule to this effect, it is common for the master to read and write on a periodic time base (polling), rather than sending and requesting data only when it is needed. In this manner, PLC register data is ensured to be valid and consistent as a representation of the slave device's status.

Wiring

Modbus RTU uses the standard RS-232 or RS-485 physical layer.

RS-232 is a point-to-point communications scheme, and as such the largest possible network would consist of a single slave drive. Please note that even though it will be the only device on the "network", it will still require an address. This address may be an integer value from 1-32, and is set through the *ST Configurator* software during initial configuration.

For drives with RS-485 communications, there are a few things to consider.

It is possible to use 2-wire RS-485 for operational communication over Modbus, however 4-wire RS-485 is required for use with all Applied Motion configuration and programming software (i.e., *ST Configurator* and *Q Programmer*). As such, we recommend that all RS-485 networks be constructed using the 4-wire method.

Be sure to consult your drive's hardware manual for specific wiring details.

Drive configuration

First, download and install the most recent version of the *ST Configurator* software from the Applied MOtion Products website (www.applied-motion.com). For RS-485 drives you will also need a USB-to-RS-485 converter, such as the USB-COMi-M module available from AMP. Consult your drive's manual for wiring and general communication instructions.

1. Launch *ST Configurator* software and select the appropriate COM port.

2. Apply power to the drive. Note, perform this step with only a single drive on the network. Each drive must be configured individually before it may be used on a network.

3. *ST Configurator* will automatically identify the drive, but will not upload all parameters. Click "Upload from Drive". This will ensure that the software accurately reflects the current state of the drive's configuration parameters.

4. Click "Motion & I/O"

5. Specify the drive's node address and baud rate.

6. For drives with Flex/IO, confirm the desired I/O settings. Click OK to return to the main screen.

7. Set any desired motor and encoder configurations your application may require.

8. Click Download to Drive to save the settings.

9. Optionally, launch *Q Programmer* to write and download a program for this axis. The drive may be configured to launch the program automatically upon powerup, or to wait for the QX command to be sent from the Modbus master.

10. Remove power from the drive and install into the main machine network.

11. Repeat for all other slave nodes, taking care to ensure each has a unique node address.

Note, the drive must be rebooted to switch between Modbus and configuration modes.

Drive Behavior

An extensive list of registers has been made available, allowing the user to monitor or change every detail of the drive's status It is also possible to send commands to a specific register, mimicking the behavior of our proprietary SCL command set. The capability allows a PLC to have unparalleled control over the drive's behavior at runtime.

The drive will respond to the following Modbus function codes:

3	Read Holding Registers
4	Read Input Registers
6	Write Single Register
16	Write Multiple Registers

Monitoring

See the Register Map table for details on specific data that can be monitored and written in this manner.

Sending Commands

The Command Opcode register, 40125, is designated to receive encoded SCL commands via Modbus. Many SCL commands have been made available in this manner, and will allow the user full control over the motion capabilities of the drive.

SCL command encoding details can be found in the Modbus Register Table.

Please refer to the Host Command Reference for details of the functionality of these registers and commands.

Examples

Point-to-Point Moves

Four pieces of data are required to fully define a point-to-point move. Acceleration, Deceleration, Distance, and Velocity. Followed of course by the move command itself. To command such a move over Modbus, we must first write to the aforementioned control registers, then send the command.

Now it's time for the command itself. The most common point-to-point move is the Feed to Length which

Function	Register	Value	Notes
Acceleration	40028	100	Units: 10 RPM/sec
Deceleration	40029	100	Units: 10 RPM/sec
Velocity	40030	240	Units: 0.25 RPM
Distance	40031..32	0x34	This register is affected by Endianness setting

uses the SCL command FL. In Modbus we do the following:

The drive will execute the move command immediately, pulling the relevant parameter data from the above registers. It is immediately ready to accept another command. Note, it is not necessary to send parameter data

Function	Register	Value	Notes
FL Command	40125	102	0x66 (Hexadecimal) is equal to 102

with each command, unless this data changes between moves. For example, to repeat a move simply send the command again, leaving the relevant paramter data unchanged.

Launch a Q Program

It is possible to execute a stored Q program over Modbus using the QX command (opcode 0x78). This command requires a value to be written to Parameter 1, the first of 5 registers set aside for command-dependant parameter data.

To launch the program at segment 1, we would use the SCL command QX1. Over Modbus, the following procedure applies.

First, configure the parameter data (only Parameter 1 is used by this command).

Function	Register	Value	Notes
Parameter 1	40126	1	Q segment to execute. Integer: 1..12

Now we can send the QX command itself.

Function	Register	Value	Notes
QX Command	40125	120	0x78 (Hexadecimal) is equal to 102

Note, to stop a command or interrupt a Q program, either the SK or SKD commands may be used.

Function	Register	Value	Notes
SK Command	40125	225	0xE1 (Hexadecimal) is equal to 225

Function	Register	Value	Notes
SKD Command	40125	226	0xE2 (Hexadecimal) is equal to 226

Modbus Register Table				
Register	**Access**	**Data Type**	**SCL Register**	**Description**
40001	Read	SHORT	Alarm Code (AL)	f
40002	Read	SHORT	Status Code (SC)	s
40003	Read	SHORT	Immediate Expanded Inputs (IS)	y
40004	Read	SHORT	Driver Board Inputs (ISX)	i
40005..6	Read	LONG	Encoder Position (IE, EP)	e
40007..8	Read	LONG	Immediate Absolute Position	I
40009..10	Write	LONG	Absolute Position Command	P
40011	Read	SHORT	Immediate Actual Velocity (IV0)	v
40012	Read	SHORT	Immediate Target Velocity (IV1)	w
40013	Read	SHORT	Immediate Drive Temperature (IT)	t
40014	Read	SHORT	Immediate Bus Voltage (IU)	u
40015..16	Read	LONG	Immediate Position Error (IX)	x
40017	Read	SHORT	Immediate Analog Input Value (IA)	a
40018	Read	SHORT	Q Program Line Number	b
40019	Read	SHORT	Immediate Current Command (IC)	c
40020..21	Read	LONG	Relative Distance (ID)	d
40022..23	Read	LONG	Sensor Position	g
40024	Read	SHORT	Condition Code	h
40025	Read	SHORT	Analog Input 1 (IA1)	j
40026	Read	SHORT	Analog Input 2 (IA2)	k
40027	Read	SHORT	Command Mode (CM)	m
40028	R/W	SHORT	Point-to-Point Acceleration (AC)	A
40029	R/W	SHORT	Point-to-Point Deceleration (DE)	B
40030	R/W	SHORT	Velocity (VE)	V
40031..32	R/W	LONG	Point-to-Point Distance (DI)	D
40033..34	R/W	LONG	Change Distance (DC)	C
40035	R/W	SHORT	Change Velocity (VC)	U
40036	Read	SHORT	Velocity Move State	n
40037	Read	SHORT	Point-to-Point Move State	o
40038	Read	SHORT	Q Program Segment Number	p
40039	Read	SHORT	Average Clamp Power (regen)	r
40040	Read	SHORT	Phase Error	z
40041..42	R/W	LONG	Position Offset	E
40043	R/W	SHORT	Miscellaneous Flags	F
40044	R/W	SHORT	Current Command (GC)	G
40045..46	R/W	LONG	Input Counter	I
40047	R/W	SHORT	Jog Accel (JA)	
40048	R/W	SHORT	Jog Decel (JL)	
40049	R/W	SHORT	Jog Velocity (JS)	J
40050	R/W	SHORT	Accel/Decel Current (CA)	
40051	R/W	SHORT	Running Current (CC)	N
40052	R/W	SHORT	Idle Current (CI)	

40053	R/W	SHORT	Steps per Revolution	R
40054	R/W	SHORT	Pulse Counter	S
40055	R/W	SHORT	Time Stamp	W
40056	R/W	SHORT	Analog Position Gain (AP)	X
40057	R/W	SHORT	Analog Threshold (AT)	Y
40058	R/W	SHORT	Analog Offset (AV	Z
40059..60	R/W	LONG	Accumulator	0
40061..62	R/W	LONG	User Defined	1
40063..64	R/W	LONG	User Defined	2
40065..66	R/W	LONG	User Defined	3
40067..68	R/W	LONG	User Defined	4
40069..70	R/W	LONG	User Defined	5
40071..72	R/W	LONG	User Defined	6
40073..74	R/W	LONG	User Defined	7
40075..76	R/W	LONG	User Defined	8
40077..78	R/W	LONG	User Defined	9
40079..80	R/W	LONG	User Defined	:
40081..82	R/W	LONG	User Defined	;
40083..84	R/W	LONG	User Defined	<
40085..86	R/W	LONG	User Defined	=
40087..88	R/W	LONG	User Defined	>
40089..90	R/W	LONG	User Defined	?
40091..92	R/W	LONG	User Defined	@
40093..94	R/W	LONG	User Defined	[
40095..96	R/W	LONG	User Defined	\
40097..98	R/W	LONG	User Defined	]
40099..100	R/W	LONG	User Defined	^
40101..102	R/W	LONG	User Defined	_
400103..104	R/W	LONG	User Defined	`
40105	R/W	SHORT	Brake Release Delay	
40106	R/W	SHORT	Brake Engage Delay	
40107	R/W	SHORT	Idle Current Delay	
40108	R/W	SHORT	Hyperbolic Smoothing Gain	
40109	R/W	SHORT	Hyperbolic Smoothing Phase	
40110	R/W	SHORT	Analog Filter Gain	
40111..124			(reserved)	
40125	R/W	SHORT	Command Opcode	
40126	R/W	SHORT	Parameter 1	
40127	R/W	SHORT	Parameter 2	
40128	R/W	SHORT	Parameter 3	
40129	R/W	SHORT	Parameter 4	
40130	R/W	SHORT	Parameter 5	

SCL Command Encoding Table							
Function	**SCL**	**Opcode**	**Parameter 1**	**Parameter 2**	**Parameter 3**	**Parameter 4**	**Parameter 5**
Alarm Reset	AX	0xBA	0	0	0	0	0
Start Jogging	CJ	0x96	0	0	0	0	0
Stop Jogging	SJ	0xD8	0	0	0	0	0
Encoder Function	EF	0xD6	0,1,2 or 6	0	0	0	0
Encoder Position	EP	0x98	Position		0	0	0
Feed to Double Sensor	FD	0x69	I/O Point 1	Condition 1	I/O Point 2	Condition 2	0
Follow Encoder	FE	0xCC	I/O Point	Condition	0	0	0
Feed to Length	FL	0x66	0	0	0	0	0
Feed to Sensor with Mask Distance	FM	0x6A	I/O Point	Condition	0	0	0
Feed and Set Output	FO	0x68	I/O Point	Condition	0	0	0
Feed to Position	FP	0x67	0	0	0	0	0
Feed to Sensor	FS	0x6B	I/O Point	Condition	0	0	0
Feed to Sensor with Safety Distance	FY	0x6C	I/O Point	Condition	0	0	0
Jog Disable	JD	0xA3	0	0	0	0	0
Jog Enable	JE	0xA2	0	0	0	0	0
Motor Disable	MD	0x9E	0	0	0	0	0
Motor Enable	ME	0x9F	0	0	0	0	0
Seek Home	SH	0x6E	I/O Point	Condition	0	0	0
Set Position	SP	0xA5	Position		0	0	0
Filter Input	FI	0xC0	I/O Point	Filter Time	0	0	0
Filter Select Inputs	FX	0xD3					
Step Filter Freq	SF	0x06	Freq	0	0	0	0
Analog Deadband	AD	0xD2	0.001 V	0	0	0	0
Alarm Reset Input	AI	0x46	Function ('1'..'3')	I/O Point	0	0	0
Alarm Output	AO	0x47	Function ('1'..'3')	I/O Point	0	0	0
Analog Scaling	AS	0xD1					
Define Limits	DL	0x42	1..3	0	0	0	0
Set Output	SO	0x8B	I/O Point	Condition	0	0	0
Wait for Input	WI	0x70	0	0	0	0	0
Queue Load & Execute	QX	0x78	1..12	0	0	0	0
Wait Time	WT	0x6F	0.01 sec	0	0	0	0
Stop Move, Kill Buffer	SK	0xE1	0	0	0	0	0
Stop Move, Kill Buffer, Normal Decel	SKD	0xE2	0	0	0	0	0

Notes:

- 32-bit values may be either big-endian or little-endian. Endianness is determined by Bit 6 of the PR command word. It is recommended that this setting be configured with the *ST Configurator™* software.
- Consult the I/O Encoding Table for details on parameter data for commands referencing I/O.

IO Encoding Table

Useful ASCII values for IO commands. Character values shown are taken from standard SCL command nomenclature. For example, a Feed to Sensor command example may utilize the falling edge of input 1 and appear as FS1F. To encode this command the user would select 0x31 for the character '1' and 0x46 for the character 'F'.

Character	hex code	
'1'	0x31	input 1 or output 1
'2'	0x32	input 2 or output 2
'3'	0x33	input 3 or output 3
'4'	0x34	input 4 or output 4
'L'	0x4C	low state (closed)
'H'	0x48	high state (open)
'R'	0x52	rising edge
'F'	0x46	falling edge

Applied Motion Products

404 Westridge Drive

Watsonville, CA 95076

USA

tel / 831-761-6555

fax / 831-761-6544

www.applied-motion.com

920-0072 Rev D
8/1/2014